REMARKS/ARGUMENTS

Amendments were made to the specification to correct errors and to clarify the specification. No new matter has been added by any of the amendments to the specification.

Claims 1-33 are pending in the present application. With this amendment, claims 5-6, 11, 16-17, 22, 27-28, and 33 have been canceled; and claims 1-2, 4, 7-8 ,12-13, 15, 18-19, 23-24, 26, and 29-30 have been amended. Reconsideration of the claims is respectfully requested.

I. Objections to the Specification

The Examiner objected to the specification because of several informalities. The specification has been amended to correct these informalities. Therefore, these objections have been overcome and should be withdrawn.

In addition, Applicants have amended the specification to cancel: "and transmission-type media such as digital and analog communications links."

II. 35 U.S.C. § 101

The Examiner has rejected claims 23-33 under 35 U.S.C. § 101 as being directed towards non-statutory subject matter.

The Examiner states: "Claim 23: recites *"a computer program product"* that may include "transmission-type media", i.e., signal - a form of energy (Specification, page 14: lines 13 - 18; *"...the present invention applies equally regardless of the particular type of signal bearing media actually used to carry out the distribution.. ."*)."

Applicants have amended claim 23 to recite "recorded on a computer-readable medium". In addition, Applicants have amended the specification to cancel: "and transmission-type media such as digital and analog communications links." Applicants believe the rejection of the claims has been overcome and should be withdrawn.

Therefore, the rejections under 35 U.S.C. §101 have been overcome.

III. 35 U.S.C. § 112, Second Paragraph

The Examiner has rejected claims 5, 16 and 27 under 35 U.S.C. § 112, second paragraph, as being indefinite for falling to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Applicants have canceled claims 5, 16, and 27. Therefore, this rejection should be withdrawn.

IV. <u>35 U.S.C. § 103(a), Obviousness</u>

The examiner has rejected claims 1-33 under 35 U.S.C. § 103(a) as being unpatentable over *Jones*, et al., <u>Method and System for Dynamic Proxy Classes</u>, (Patent No. 6,877,163, dated April 5, 2005) in view of *McIntyre*, <u>Method and Apparatus for Determining Compatibility of Parent Classes in an Object Oriented Environment Using Versioning</u>, (Patent No. 6,415,435, dated July 2, 2002). This rejection, as it might be applied to the claims as amended, is respectfully traversed.

Applicants' independent claims, claims 1, 12, and 23, recite similar features. Claim 1 is representative of these features. Claim 1 recites: A method in a data processing system that implements an object-oriented software environment for translating method calls to version-specific method calls, said method comprising the steps of: providing an interface to an underlying object, applications utilizing said interface to communicate with said underlying object, said interface being separate from said underlying object; generating a plurality of version-specific underlying objects, each one of said version-specific underlying objects being a different version of said underlying object; generating a plurality of translation objects, each one of said plurality of translation objects for communicating between said interface and a different one of said version-specific underlying objects; each one of said plurality of translation objects including a proxy instance and an invocation handler instance; receiving, by a particular one of said plurality of translation objects, a particular interface method call that was invoked on said interface, said particular one of said plurality of translation objects for communicating between said interface and a particular one of said version-specific underlying objects, said particular interface method call including a name and formal parameters; translating, by a particular invocation handler instance that is included in said particular one of said plurality of translation objects, said particular interface method call by determining a version-specific method call that corresponds to said particular interface method call using said name and said formal parameters of said particular interface method call; invoking, by said particular invocation handler instance, said version-specific method call; wherein, said plurality of translation objects are used to translate interface method calls that were invoked on said interface to corresponding version-specific method calls for each version of said underlying object; and generating said plurality of translation objects from a single proxy class and a single invocation handler class, wherein the same proxy class and invocation handler class are used to generate said plurality of translation objects.

The Examiner states that *Jones* teaches generating a plurality of version-specific underlying objects, each one being a different underlying object by teaching event generators. Applicants respectfully disagree.

Jones teaches different event generators, e.g. a device driver and a windows manager. *Jones* does not teach different versions of one event generator. For example, *Jones* does not teach different versions of a device driver.

Although the Examiner states that version-specific objects are simply objects that belong to a specific type of object and that are fundamentally the same but slightly different in some supported versions, *Jones* does not teach version-specific objects. The Examiner does not refer to any particular section in *Jones* that teaches version-specific objects.

The Examiner states that *Jones* teaches generating a translation object by teaching a proxy class. Applicants now claim generating a plurality of translation objects, each one of said plurality of translation objects for communicating between said interface and a different one of said version-specific underlying objects. Neither *Jones* nor *McIntyre* teaches generating a plurality of translation objects, each one of said plurality of translation objects for communicating between said interface and a different one of said version-specific underlying objects, where the plurality of translation objects are generated from a single proxy class and a single invocation handler class.

Furthermore, the combination of *Jones* and *McIntyre* does not teach this feature. *McIntyre* merely teaches the existence of child and parent versions. Combining the idea of child and parent versions with a proxy class does not teach generating a <u>plurality</u> of translation objects, where each translation object is for communicating between the interface and a <u>different one</u> of said version-specific underlying objects.

Applicants' claims now describe receiving, by a particular one of said plurality of translation objects, a particular interface method call that was invoked on said interface, said particular one of said plurality of translation objects for communicating between said interface and a particular one of said version-specific underlying objects, said particular interface method call including a name and formal parameters. Combining the idea of child and parent versions *Jones* does not teach these features.

Applicants' claims now describe translating, by a particular invocation handler instance that is included in said particular one of said plurality of translation objects, said particular interface method call by determining a version-specific method call that corresponds to said particular interface method call using said name and said formal parameters of said particular interface method call.

Regarding original claim 8, the Examiner stated that *Jones* teaches an invocation handler is

[illegible] according to the specified interfaces as parameters at the client. Applicants claim a particular